Knowledge base and database representation for intelligent concurrent design

D. Xue*, S. Yadav, D.H. Norrie

Department of Mechanical and Manufacturing Engineering, The University of Calgary, Calgary, Alberta, Canada T2N 1N4

Abstract

This article presents methods for modeling knowledge base and database for intelligent concurrent design, including mathematical formulation and system development. Four issues are addressed in this research: (1) modeling product life-cycle aspects using aspect primitives called features, (2) generating product life-cycle aspects using a knowledge-based system, (3) maintaining consistency of product life-cycle aspects using the relations among these aspects, and (4) identifying the optimal design considering relevant product life-cycle aspects. A case study example is given in the end to demonstrate the effectiveness of the introduced methods. © 1999 Elsevier Science Ltd. All rights reserved.

Keywords: Concurrent design; Knowledge base; Database

1. Introduction

1.1. Concurrent design

Development of a product undergoes a sequence of life-cycle phases including design, production planning, manufacturing, inspection, and so on. In the past, these product development activities were organized in a sequential manner: from design to other downstream life-cycle phases. As the design engineers tend to generate designs with excellent functional performance, these designs usually require excessive manufacturing efforts, thereby leading to high production cost.

To reduce the gap between design and other downstream product development aspects, researches on concurrent design were initiated [1,2]. Concurrent design is an approach which incorporates the considerations in the downstream product development phases, including manufacturing, assembly, maintenance, and so on, into the design phase for producing a design with the best overall product life-cycle performance. In the concurrent design, as evaluations to the downstream product development aspects are carried out simultaneously at the early design stage, this approach can reduce the number of costly redesigns and shorten product development lead time.

The research on computer-aided concurrent design was started by the pioneer work on feature recognition [3,4]. Feature recognition approach aims at extracting geometric shapes, such as holes and slots, to be produced by certain manufacturing processes from the CAD database for planning production process and evaluating the design from the manufacturing point of view. Due to the difficulty in feature recognition, another approach, called design by features that uses manufacturing features as primitives for modeling a design, has been employed by many researchers [5–7]. A detailed review on feature recognition and feature-based design is given in Ref. [8].

The design-for-X approach incorporates the considerations in one downstream product development life-cycle aspect into the design phase, thereby generating a design with good evaluation in the life-cycle aspect. Typical design for X methods include design for manufacturing [9,10], design for assembly [11,12], design for serviceability [13,14], design for disposal/recycle [15], etc. Many advanced computational techniques, including rule-based reasoning [16], constraint-network [17], and optimization [18], have been employed for developing automated concurrent design systems.

1.2. Previous research on intelligent concurrent design

The research on intelligent concurrent design, which was initiated by Xue [19–25], aims at developing a computer-based concurrent design environment that supports the activities in all the product development life-cycle aspects, thereby providing a theoretical and implementational
framework for the next generation CAD/CAM systems with concurrent design capability.

In this research, product life-cycle aspects are modeled by aspect primitives called aspect features, such as design features (mechanism and components, such as a gear and a shaft), and manufacturing features (geometric elements to be produced, such as a hole and a slot) [19]. Features are represented using the scheme of a product modeling language – integrated data description language (IDDL) [26,27]. A system that combines knowledge-based reasoning and optimization was introduced to automatically generate the aspect models and identify the optimal design using optimization [20]. Due to the large feature library size, a design-function based design feature coding system and a manufacturing-function based manufacturing feature coding system were developed for organizing feature library and for automatically generating design candidates and planning production process [23]. As production cost is a key measure for evaluating manufacturability, a number of cost models with different production processes and tolerance requirements were introduced [21]. An optimization model was developed for obtaining the design with the best tradeoff between functional performance and production cost [22]. A number of global optimization models were also introduced to identify the optimal design efficiently [24]. The optimization-based concurrent design approach was employed in designing a fuel cell system [25].

The research presented in this article aims at developing an intelligent concurrent design system based on the results obtained in the previous research with focus on knowledge base and database representation. The mathematical formulation for intelligent concurrent design is first introduced and their development and implementation then presented. A case study example is given in the end to demonstrate the effectiveness of the introduced methods.

2. Mathematical models for intelligent concurrent design

The following presents the mathematical models that were developed for representing product life-cycle, product realization process, data relation maintenance, and optimal design identification in intelligent concurrent design. The intelligent concurrent design system described subsequently was implemented based on these mathematical models.

2.1. Product life-cycle aspect models

A product life-cycle can be considered from different points of views, called aspects. The different product life-cycle aspects can be modeled by various aspect models, such as design aspect model, M(\(D\)), manufacturing aspect model, M(\(M\)), and so on. Each aspect model is constructed using aspect building primitives, namely aspect features [19]. An aspect feature, F(\(i\))(\(i = 1,2,\ldots,n_F\)), is a group of descriptions in the aspect model for a particular product development purpose. The superscript (P) of F(\(i\)) is used to describe the product life-cycle aspect, such as design aspect, (D), manufacturing aspect, (M), and so on. For instance, a gear-pair is a design feature which delivers a rotation-to-rotation transmission function, and a hole is a manufacturing feature to be produced by a drilling machining operation. The concept of feature in this research is an extension of the conventional feature concept that is used to define only component geometry to be produced by certain manufacturing operations. An attribute of a feature, A(\(ij\)), is a quantitative description in the aspect model and is defined as

\[
A_{ij} = A_{ij}^P(F_i^P), \quad i = 1,2,\ldots,n_F^P; \quad j = 1,2,\ldots,n_A^P.
\] (1)

For instance, nominal diameter, module, and tooth number are the three attributes of a gear.

Features and attributes are associated by their qualitative relations, R(\(F\)), and quantitative relations, R(\(A\)), respectively. These relations are defined as

\[
R_{F_l} = R_{F_l}^P(F_1^P,F_2^P,\ldots,F_{n_F}^P), \quad l = 1,2,\ldots,n_{R_F}^P, \quad (2)
\]

\[
R_{A_m} = R_{A_m}^P(A_1^P,A_2^P,\ldots,A_{n_A}^P), \quad m = 1,2,\ldots,n_{R_A}^P. \quad (3)
\]

For instance, the relation between two gear features in a gear-pair mechanism is a qualitative relation, while the relation used to calculate the nominal diameter attribute of a gear using the tooth number and module as input attributes is a quantitative one.

An aspect model is described by the collection of aspect features, F(\(P\)), attributes, A(\(P\)), and their qualitative relations, R(\(F\)), and quantitative relations, R(\(A\)), using

\[
M(\(P\)) = \{F(\(P\)), A(\(P\), R(\(F\)), R(\(A\))\}, \quad (4)
\]

where

\[
F(\(P\)) = \{F_1^P,F_2^P,\ldots,F_{n_F}^P\}, \quad (5)
\]

\[
A(\(P\)) = \{A_1^P,A_2^P,\ldots,A_{n_A}^P\}, \quad (6)
\]

\[
R(\(F\)) = \{R_{F_1}^P,R_{F_2}^P,\ldots,R_{F_{n_{R_F}}}^P\}, \quad (7)
\]

\[
R(\(A\)) = \{R_{A_1}^P,R_{A_2}^P,\ldots,R_{A_{n_{R_A}}}^P\}. \quad (8)
\]

A product, \(P\), is described by all aspect models, M, and their relations, R(\(F\)) and R(\(A\));

\[
P = \{M,R(\(F\)),R(\(A\))\}. \quad (9)
\]

where M is defined as

\[
M = \{M(\(D\)),M(\(M\)),\ldots\}. \quad (10)
\]

2.2. A product realization process model

The product life-cycle aspect models are constructed in a
Sequential manner. First, the design candidates are created to satisfy the design functions. The geometric models are then generated to describe the design details. Manufacturing operation descriptions are subsequently achieved for producing the desired geometry. This progressive nature of product development activities is represented by a product realization process model. This model is an extension of the general design theory (GDT), in which a design is considered as a process of mapping from function space to attribute space \[28,29\].

In the product realization process model, new product descriptions, \( M_0 \), at a certain product development stage are derived from the product descriptions, \( M \), at an earlier product development stage using relevant knowledge, \( K \), as described by
\[
M_0 \leftarrow K \triangleright M
\]

In this equation, \( \triangleright \) and \( \leftarrow \) are logical symbols representing the AND relation and the data derivation relation, respectively. For instance, the geometric model of a product, \( M(G) \), can be derived from the design model of the same product, \( M(D) \).

The relations among the derived data are classified into two categories: AND relations and OR relations. For instance, the three machining operation descriptions derived by
\[
\text{hole } \triangleleft K_1 \leftarrow \text{drilling } \cap \text{reaming } \cap \text{grinding }
\]

have an AND relation, while the two design candidate descriptions generated by the following two equations
\[
\begin{align*}
\text{rotation-to-rotation transmission } & \triangleleft K_2 \\
\text{gear-pair mechanism,}
\end{align*}
\]

\[
\begin{align*}
\text{rotation-to-rotation transmission } & \triangleleft K_3 \\
\text{pulley-belt-drive-pair mechanism}
\end{align*}
\]

have an OR relation.

### 2.3. A data relation maintenance model

The different product life-cycle aspect models are integrated by their relations, which include qualitative relations among features and quantitative relations among attributes. In a concurrent design, first the product realization process model is used to generate aspect models and their relations. As a description is usually derived from other descriptions using relevant knowledge, change of an earlier created description should have an influence on the derived descriptions. For instance, the machining operation descriptions in Eq. (12) are derived from the hole manufacturing feature. If the hole manufacturing feature description is removed from the database, the three derived machining operation descriptions should also be deleted. This dependent relation is described by
\[
d_j \leftarrow d_1, d_2, \ldots, d_n,
\]

where \( d_i \) could be a feature, an attribute, a qualitative relation, or a quantitative relation.

A quantitative relation among attributes can be further described by
\[
A_j \leftarrow A_1, A_2, \ldots, A_n
\]

where \( A_j \) is calculated using \( A_1, A_2, \ldots, A_n \) as input attributes.

### 2.4. An optimal concurrent design model

An optimal concurrent design model is used to identify the optimal design alternative and its parameters considering relevant life-cycle aspects. The optimization is carried out at two levels: (1) alternative optimization level, and (2) parameter optimization level [24].

A feasible alternative is described by a number of features and their attributes. As the qualitative descriptions of features remain the same for a design alternative, a feasible alternative, \( P_0 \), can therefore be described by a vector of attributes:
\[
P_0 = \{A_{i1}, A_{i2}, \ldots, A_{in}\}.
\]
The optimal parameter values considering one design alternative are obtained by constrained optimization approach given by

$$\min_{w.r.t. A_{i1}, A_{i2}, ..., A_{in}} f_i(A_{i1}, A_{i2}, ..., A_{in})$$

s.t.

$$h_j(A_{i1}, A_{i2}, ..., A_{in}) = 0, \quad j = 1, 2, ..., k_i,$$

$$g_j(A_{i1}, A_{i2}, ..., A_{in}) \leq 0, \quad j = k_i + 1, k_i + 2, ..., m_i.$$ (18)

The objective function, $f_i(A_{i1}, A_{i2}, ..., A_{in})$, is an evaluation measure of the design from a certain product life-cycle perspective.

The optimal objective function evaluation measure for the design alternative is described as $f_i(P_i^*)$. The optimal alternative is identified from all possible alternatives using

$$\min_{w.r.t. P_i^*} f_i(P_i^*),$$ (19)

where $P_i^*$ is iterated among the feasible alternatives with the optimal parameters.

3. Development of an intelligent concurrent design system

An intelligent concurrent design system was developed based on the mathematical formulation of intelligent concurrent design and previously developed prototype systems [19–25]. In the intelligent concurrent design system, product life-cycle aspects are described by aspect models, including design aspect model, manufacturing aspect model, and so on, as shown in Fig. 1. Aspect features serve as building primitives for modeling and evaluating these aspect models. Aspect models are generated through knowledge-based reasoning. Relations among aspect models are organized by data relation networks. The optimal design alternative and its parameters are identified using optimization. The system was implemented using Smalltalk, an object oriented programming language [30].

3.1. Representation of product life-cycle aspect models using feature-based modeling approach

The product life-cycle aspect models are built using aspect primitives called features, such as design features (mechanisms and components to satisfy the design functions), manufacturing features (component geometry to be produced by certain manufacturing processes), etc. In this research, a feature modeling environment was developed. Features are described at two levels, class level and instance level, corresponding to generic libraries and specific product data, respectively (Fig. 2). Instance features are generated using class features as their templates. This mechanism was implemented using object-oriented programming approach.

3.1.1. Class features

A new class feature is defined using an existing class feature as its super-class. Class features are organized in a hierarchical data structure. All the descriptions in a
super-class feature can be inherited automatically by its sub-class features. The top level class feature is a built-in class feature called Feature. Examples of class feature definitions are shown in Fig. 2. A class feature is composed of the following components:

3.1.1.1. Element features
An element feature is a feature that composes the feature being defined. For instance, the GearPair class feature, shown in Fig. 2, is composed of two gear element features. An element feature is defined by a variable and its class feature type. A variable is a string starting with '?'.' In the class feature GearPair, the two gear element features are associated with two variables: ?X and ?Y. When a class feature is used to generate an instance feature, the element features should also be created according to their class feature types. The variables, representing the element features, are used in other parts of the class feature definition. For instance, the two variables, ?X and ?Y, are used for defining feature relations and attribute relations in GearPair. The feature itself is associated with a special variable, ?self.

3.1.1.2. Attributes
An attribute is a piece of quantitative description of the feature and is described by an attribute name and an attribute value. Attributes are used in the form of attribute[feature] in other parts of the feature definition. For instance, n[?X] in Fig. 2 represents the attribute n of the feature ?X.

3.1.1.3. Feature relations
Qualitative relations among features are described by predicates. A predicate takes the form of \((x_1, x_2, \ldots, x_n)\), where \(x_1, x_2, \ldots, x_n\) are terms of this predicate represented by symbols, strings, integers, floats, variables, and attributes. Definitions and examples of the different types of terms are given in Table 1. A predicate without variable terms is called a fact.

3.1.1.4. Attribute relations
A quantitative relation among attributes is defined by a function with a number of input attributes and one output attribute. An attribute relation takes the following form

\[
\text{attribute name} := \langle \text{expression} \rangle
\]

Syntax of \(\langle \text{expression} \rangle\) follows the syntax of Smalltalk. In addition, element feature variables and attributes are allowed in the expression. Examples of attribute relations are shown in Fig. 2.

In addition to the four types of components introduced earlier, many other types of components, including constraints, two-dimensional (2D) and three-dimensional (3D) geometric descriptions, etc., can also be defined in a class feature.

3.1.2. Instance features
Instance features are generated from class features for modeling a product. When a class feature is selected for generating an instance feature, the element features defined in this class feature should also be generated as instance features. All the descriptions in the class feature and its super-class features are inherited by the generated instance feature automatically. Fig. 2 shows the instance features that are generated from class features for representing a gear-pair mechanism. The different product life-cycle aspects are modeled by instance features and their relations.

In an instance feature, the variables in its class feature definition representing element features are replaced by the names of the actually created element instance features. For example, in the instance feature gearPair1 shown in Fig. 2, the two element feature variables, ?X and ?Y, are replaced by the two element instance features, g1 and g2. Descriptions of the instance features can be modified, added, and deleted.

3.1.3. Geometric representation in features
The feature-based product life-cycle representation approach is primarily used for modeling both the qualitative

---

### Table 1

<table>
<thead>
<tr>
<th>Types</th>
<th>Definitions</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symbol</td>
<td>A sequence of characters starting with a letter</td>
<td>Gear</td>
</tr>
<tr>
<td>String</td>
<td>A sequence of characters starting and ending with &quot;&quot;</td>
<td>&quot;Hello&quot;</td>
</tr>
<tr>
<td>Integer</td>
<td>An integer number</td>
<td>25, -14</td>
</tr>
<tr>
<td>Float</td>
<td>A real number</td>
<td>1.2e-25</td>
</tr>
<tr>
<td>Variable</td>
<td>A sequence of characters starting with '?'</td>
<td>?X</td>
</tr>
<tr>
<td>Attribute</td>
<td>An attribute name symbol and a feature name symbol (or variable) linked by brackets [ and ]</td>
<td>z[?gear]</td>
</tr>
</tbody>
</table>

---

Fig. 3. Integration of feature-based product modeling and product geometry modeling.
and quantitative descriptions of the life-cycle aspects. The detailed product geometry, however, cannot be modeled by features directly. As the representation of product geometry plays an important role in modeling the product database, integration of the feature-based product modeling system with the product geometry modeling system is required. In this research, 3D Studio MAX [31] is employed as the product geometry modeling system to be integrated with the feature-based product modeling system. Relations between the two systems are maintained through the communication between the two mediators: feature-based product modeling mediator and product geometry modeling mediator, as shown in Fig. 3.

The geometry of a class feature is described as a part of the feature definition using a collection of built-in predicates. Typical built-in predicates are shown in Table 2. Features and attributes are allowed to define the terms of these predicates. When instance features are generated using class features as templates, the geometric descriptions are also inherited by these instance features. The feature-based product modeling mediator extracts the 3D geometric descriptions from the instance features. The product geometry modeling mediator further translates these descriptions into the 3D geometric data in 3D Studio MAX. Change in one system should be updated to the other through communication between the two mediators.

For example, an assembly, as shown in Fig. 4(a), is composed of two components: a base and a shaft. The two components and the assembly are first described by three class features: Base, Shaft, and BaseShaft, respectively, as shown in Fig. 4(b). In the Base class feature, the geometry of a base is defined as the result of a subtraction Boolean operation to a box and a cylinder. The box is created by defining a 2D rectangle primitive and extruding this primitive. The cylinder is created by defining a 2D circle primitive and extruding this primitive. In the Shaft class feature, the geometry of the shaft is defined as the result of a union Boolean operation to two cylinders. In each of these two class features, the final geometric object name is described by the variable ?self.

Three instance features, base1, shaft1, and baseShaft1, are generated from the three class features, respectively.

The instantiated 3D geometric descriptions in these three instance features are illustrated in Fig. 4(c). The attribute values and relations representing the positions of these objects are defined as:

\[
\begin{align*}
\text{locX[base]} & := 0.0 \\
\text{locY[base]} & := 0.0 \\
\text{locZ[base]} & := 0.0 \\
\text{locX[shaft1]} & := \text{locX[base]} \\
\text{locY[shaft1]} & := \text{locY[base]} \\
\text{locZ[shaft1]} & := \text{height[base]} = \text{height1[shaft1]}
\end{align*}
\]

When the 3D geometric descriptions in the instance features are used to generate the input data for the 3D Studio MAX, all the attribute names are replaced by their attribute values. The temporary geometric object names defined in these instance features are associated with the unique instance feature names for generating intermediate geometric objects in the 3D Studio MAX environment. The generated 3D geometric descriptions for 3D Studio MAX are shown in Fig. 4(d).

### 3.2. Automated generation of product life-cycle aspect models using a knowledge-based system approach

The product realization process is described by an AND/OR graph, as shown in Fig. 5. A node in this graph is described by either a feature, an attribute, a feature relation, or an attribute relation. The AND and OR relations among the sub-nodes can be achieved using the knowledge as described in Eqs. (12), (13), and (14).

A knowledge base system was developed for generating the AND/OR graph through knowledge-based reasoning. In this system, knowledge is described by rules. Rules are organized in rule-base. Fig. 6 shows an example rule-base. The idea for organizing rules in groups follows the concept introduced for the intelligent integrated interactive CAD (IIICAD) system [26,27].

A rule-base is defined by a rule-base name and a collection of rules. Each rule description is composed of the rule

<table>
<thead>
<tr>
<th>Categories</th>
<th>Term 1</th>
<th>Term 2</th>
<th>Term 3</th>
<th>Term 4</th>
<th>Term 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D primitive generation</td>
<td>circle (name)</td>
<td>(radius)</td>
<td>(length)</td>
<td>(width)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>rectangle (name)</td>
<td>(name)</td>
<td>(length)</td>
<td>(width)</td>
<td>(height)</td>
</tr>
<tr>
<td>3D primitive generation</td>
<td>box (name)</td>
<td>(name)</td>
<td>(length)</td>
<td>(width)</td>
<td>(height)</td>
</tr>
<tr>
<td></td>
<td>cylinder (name)</td>
<td>(name)</td>
<td>(radius)</td>
<td>(height)</td>
<td></td>
</tr>
<tr>
<td>2D primitives to 3D primitives</td>
<td>extrude (2D name)</td>
<td>(3D name)</td>
<td>(height)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>lathe (2D name)</td>
<td>(3D name)</td>
<td>(degree)</td>
<td>(axis)</td>
<td></td>
</tr>
<tr>
<td>Transformation operations</td>
<td>move (name)</td>
<td>(x)</td>
<td>(y)</td>
<td>(z)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>rotate (name)</td>
<td>(θx)</td>
<td>(θy)</td>
<td>(θz)</td>
<td></td>
</tr>
<tr>
<td>Boolean operations</td>
<td>union (name 1)</td>
<td>(name 2)</td>
<td>(name)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>subtraction (name 1)</td>
<td>(name 2)</td>
<td>(name)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Organization of objects</td>
<td>group (name)</td>
<td>(node 1)</td>
<td>(node 2)</td>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>ungroup (name)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Fig. 4. 3-D geometric representation for an assembly with two components.
name and the rule itself. A rule takes the form of IF–THEN data structure, representing a piece of cause–result knowledge. Both the IF and the THEN part of a rule are represented by a number of patterns linked with logical-and (&). A pattern is described by a predicate using the form of \( x_1; x_2; \ldots; x_n \), where \( x_1, x_2, \ldots, x_n \) are terms. Terms are described by symbols, numbers, variables, and attributes, as introduced in Section 3.1. A predicate without variable terms in the database is called a fact. The condition part and result part of a rule are used for matching, creating, deleting, and modifying the data in aspect models, including features, attributes, qualitative relations among features (facts), and quantitative relations among attributes (functions). In the rule-base shown in Fig. 6, the built-in predicates feature-Type, assertFeature, and assertFunction are used for checking the class type of an instance feature, adding an instance feature, and adding a function, respectively.

To improve the reasoning efficiency, only part of the data in the database and part of the knowledge in the knowledge base are selected for inference. In this work, an instance feature is such a database unit for inference. An instance feature is composed of element features, attributes, qualitative relations among features, and quantitative relations among attributes. The knowledge used in inference is a collection of selected rule-bases. Therefore, each instance feature is associated with a number of rule-bases. This idea is illustrated in Fig. 7.

The knowledge-based product modeling starts with selecting an instance feature as the active instance feature and selecting the rule-bases used for this active instance feature. When a number of rule-bases are selected, all the rules in these rule-bases should be registered in the active instance feature. The inference is carried out first by matching the condition parts of all the registered rules with the active instance feature database. If multiple rules are matched, the best rule is then selected and the result part of this rule executed. In this research, the first matched rule is considered as the best rule to be fired.

The AND/OR graph of product realization process is generated by knowledge-based reasoning. When condition part of a rule matches with the database, the data created by executing the result part of this rule have an AND relation. If several rules match with the same data, the data generated by these rules have an OR relation. Generation of AND/OR trees using rules is illustrated in Fig. 8.

### 3.3. Organization of qualitative and quantitative relations among product aspect models

Product life-cycle aspects are modeled by features, attributes, qualitative relations among features, and quantitative relations among attributes. As these aspect models represent the same product from different perspectives, change in one aspect should be updated to the other aspects automatically using the relations among aspects models. In this research, two associated networks, feature relation network and attribute relation network, were developed for modeling and maintaining the qualitative and quantitative relations. This mechanism is a further improvement of a previously

**Rule-Base:** rotationToRotationFunctionDesign
- Rule: pulleyBeltDrivePair
  - IF (featureType, ?x, RotationToRotation) THEN (assertFeature, ?y, PulleyBeltDrivePair).
- Rule: gearPair
  - IF (featureType, ?x, RotationToRotation) THEN (assertFeature, ?y, GearPair).
- Rule: speedRatioDescription
  - IF (rotationToRotation, ?x, ?y) THEN (assertFunction, i[?x]=n(?y)/n(?x)) & (assertFunction, i[?y]=n(?x)/n(?y)).

![Fig. 5. An AND/OR graph of product realization process.](image5)

![Fig. 6. A rule-base.](image6)
developed quantitative intelligent system mechanism [20] for the newly introduced feature representation scheme. The feature relation network is either generated manually or by the knowledge-based system. This network is composed of instance features and their dependent relations, representing the AND/OR tree of the product realization process. Feature descriptions, including element features, attributes, qualitative relations among element features, and quantitative relations among attributes, are associated with these instance features. A feature relation network representing the two design alternatives is shown in Fig. 9.

From an AND/OR tree of a feature relation network, product realization process alternatives, which are described by a number of instance features and their descriptions, can be obtained. An alternative is identified using the following steps:

1. From the AND/OR tree, select the instance feature that represents the interested product development requirement.
2. If a selected instance feature has descendant instance features with an AND relation, all these descendant instance features should be selected.
3. If a selected instance feature has descendant instance
Fig. 9. Organization of qualitative and quantitative relations.
features with an OR relation, only one of these descendant instance feature should be selected.

Steps 2 and 3 are carried out continuously until no selection is required.

In the example shown in Fig. 9, two design alternatives described by instance features of (1) mechanism1, wormGear1, rotationToRotation1, gearPair1, shaft1, worm1, gear1, shaft2, gear2, gear3, and shaft3, and (2) mechanism1, wormGear1, rotationToRotation1, pulleyBeltDrivePair1, shaft1, worm1, gear1, shaft2, pulley1, pulley2, and shaft3, are obtained.

For each product realization process alternative, the relations among attributes then form another network, called attribute relation network. An attribute relation network is composed of two types of nodes: attribute and function. A function node is linked with a number of input attribute nodes and one output attribute node. The two attribute relation networks, generated from the two design alternatives, are shown in Fig. 9(c).

3.4. Identification of the optimal design using a multi-level optimization approach

Identification of the optimal design is carried out at two levels: (1) parameter optimization level, and (2) alternative optimization level. First, feasible design alternatives are obtained from the AND/OR graph. Each alternative is described by part of the data in the AND/OR graph including features, attributes, and their qualitative and quantitative relations. The optimal parameter values considering one alternative can be obtained at the parameter optimization level using constrained optimization search. The optimal alternative is achieved from all the feasible alternatives at the alternative optimization level. Many advanced optimization methods, including genetic algorithm and simulated annealing, can be employed for improving the search efficiency and quality [24].

Objective function is selected based on the requirements in concurrent design. In the previous research for improving manufacturability, three types of objective functions have been introduced [22]. They are: (1) production cost function, (2) production time function, and (3) combined cost and time functions. Suppose the production cost and time in relation to the i-th alternative are represented by $C_i(A_{i1}, A_{i2}, \ldots, A_{in})$ and $T_i(A_{i1}, A_{i2}, \ldots, A_{in})$, respectively. Then, the objective function $f_i(A_{i1}, A_{i2}, \ldots, A_{in})$ in Eq. (18) can be represented by one of the three functions shown in Table 3. The $\alpha_i$ and $\beta_i$ are the weighting factors between 0 and 1 for indicating the importance of production cost and time in manufacturability evaluation.

<table>
<thead>
<tr>
<th>Manufacturability considerations</th>
<th>Objective functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Production cost only</td>
<td>$C_i(A_{i1}, A_{i2}, \ldots, A_{in})$</td>
</tr>
<tr>
<td>Production time only</td>
<td>$T_i(A_{i1}, A_{i2}, \ldots, A_{in})$</td>
</tr>
<tr>
<td>Both production cost and time</td>
<td>$\alpha_i C_i(A_{i1}, A_{i2}, \ldots, A_{in}) + \beta_i T_i(A_{i1}, A_{i2}, \ldots, A_{in})$</td>
</tr>
</tbody>
</table>

4. System implementation

The intelligent concurrent design system was developed based on the methods introduced earlier and the architecture of this is shown in Fig. 10.

The system was implemented using Smalltalk [30] and 3D Studio MAX [31]. Users are classified into two different groups: knowledge modeling users and product modeling users. The knowledge modeling users are responsible for modeling and maintaining generic libraries, including class features and rule-bases using two interface windows called Class Feature Browser and Rule-Base Browser, respectively. The product modeling users build the product database by generating instance features using an interface window called Instance Feature Browser. Instance features
Fig. 11. A snapshot of the developed system.

(a) Design alternative: c1

(b) Design alternative: c2

(c) AND/OR tree of the product realization process

f: functional requirement  th: thread hole  t: thread  d: drilling  b: boring

h: hole  p: process  r: reaming  tg: threading

Fig. 12. An example for identifying the optimal design.
are generated from class features either manually or through rule-based inference. The data relation maintenance mechanism and the optimal design mechanism were implemented as two modules to be executed in the instance feature browser. The 3D geometric descriptions in the instance features are extracted and translated into the 3D Studio MAX descriptions. A snapshot of the developed system is shown in Fig. 11.

5. An intelligent concurrent design example

A simple case study example is given in this section to show how the optimal design alternative and its parameters are identified considering manufacturability and using the introduced methods. The problem is to design the internal thread features at the two ends of a shaft. This functional requirement can be satisfied by either designing the two internal thread features on the surfaces of two holes (alternative c1 in Fig. 12(a)), or designing the two internal thread features on the surface of one hole (alternative c2 in Fig. 12(b)). An internal thread feature is created by first producing the hole, and then the thread. A hole is machined by either (1) drilling → reaming process, or, (2) drilling → boring process. A thread is produced by threading machining operation. The AND/OR graph of the product realization process, which is generated through knowledge-based reasoning, is illustrated in Fig. 12(c).

In this example, the length attributes of the two thread features, \( L_1 \) and \( L_2 \), are selected as the variables at the parameter optimization level. Both \( L_1 \) and \( L_2 \) should be longer than 30 mm to satisfy the design requirements. The length of the hole in case \( c_1 \) should be 5 mm longer than the length of the thread. Production costs for producing the internal thread features regarding different machining operations are functions of the cutting length, as shown in Table 4. The two design candidates, \( c_1 \) and \( c_2 \), can be produced by different manufacturing process alternatives, as shown in Table 5. For the ease of explanation, only the bottom nodes representing the machining operations are used for describing these alternatives.

The total production cost is selected as the objective function to be minimized for this problem. Using the cost models introduced in Table 4, the total cost is a function of the two selected variable attributes for each alternative.

The optimal parameters regarding one alternative are identified using constrained optimization. For instance, optimization for the first alternative is formulated as:

\[
\begin{align*}
\text{min} & \quad \text{cost}[d1] + \text{cost}[r1] + \text{cost}[tg1] + \text{cost}[d3] \\
& + \text{cost}[r3] + \text{cost}[tg2] \\
= & \quad 0.03(L1[tg1] + 5) + 0.06(L1[tg1] + 5) + 0.09 \times L1[tg1] \\
& + 0.03(L2[tg2] + 5) + 0.06(L2[tg2] + 5) + 0.09 \\
& \times L2[tg2] \\
\text{subject to} & \quad L1[tg1] \geqslant 30 \\
& \quad L2[tg2] \geqslant 30
\end{align*}
\]

The optimal parameter values are achieved as:

\( L1^*[tg1] = 30 \)

\( L2^*[tg2] = 30 \)

and the total cost is calculated as $11.7. In the same way, the minimum costs for other alternatives are calculated as shown in Table 5. The optimal alternative is then identified from all the feasible alternatives.

Modeling of the fuel cell system introduced in [25] was also tested to verify the effectiveness of the developed intelligent concurrent design system. The fuel cell system consists of four modules: fuel cell stack module, hydrogen supply module, air supply module, and cooling module. These modules are modeled by instance features. Two attributes are selected as design variables for optimization: (1) active fuel cell stack intersection area, and (2) air stoichiometric ratio. The fuel cell system is evaluated by (1) maximum net system power output, (2) average efficiency, and (3) total production cost. In addition, a building product design system was also developed for a local manufacturing company – Gienow Building Products Ltd.

6. Conclusions

In this research, a number of mathematical models were introduced for representing the intelligent concurrent design. An intelligent concurrent design system was
developed and implemented based on these mathematical models. The advantages of this system are summarized as follows:

1. By introducing features as primitives for representing product life-cycle aspects, the efficiency of modeling product database can be improved considerably.
2. The knowledge-based system can further improve the product modeling efficiency. In addition, the AND/OR tree generated by knowledge-based reasoning is effective to model the product realization process.
3. The data relation networks provide an efficient mechanism for maintaining the consistency among product life-cycle aspect models.
4. The multi-level optimal design model is efficient in identifying the optimal design alternative and its parameters from the AND/OR tree of the product realization process model.

The introduced method has greatly improved the product modeling efficiency. This approach also provides a platform for developing the next generation CAD systems with concurrent design capabilities.

Acknowledgements

The authors would like to thank Profs. H. Yoshikawa, T. Tomiyama at the University of Tokyo and Prof. Z. Dong at the University of Victoria for their support in developing the product description language IDDL and the early version of the intelligent concurrent design system. The authors would also like to thank Z. Feng for his early effort in implementing this intelligent concurrent design system. The financial support from the Natural Sciences and Engineering Research Council (NSERC) of Canada and Gienow Building Products Ltd. is also acknowledged.

References

D. Xue is an assistant professor in the Department of Mechanical and Manufacturing Engineering, University of Calgary. He received his PhD and MSc in Precision Machinery Engineering from the University of Tokyo in 1992 and 1989, respectively, and his BSc in Precision Instrument Engineering from Tianjin University in 1985. His research interests include design for manufacturing and feature-based concurrent design, engineering optimization, and knowledge base and database representation for intelligent CAD/CAM systems. He is a member of ASME, SME, AAAI, and IPSJ.

D.H. Norrie currently holds the Nortel Chair in Intelligent Manufacturing at The University of Calgary, Alberta, Canada. Formerly, he was the head of the Division of Manufacturing Engineering at the University of Calgary. He is also Professor of Mechanical and Manufacturing Engineering and Adjunct Professor of Computer Science at the same institution. His research interests are in Intelligent Systems and, in particular, in Multi-Agent Applications in Manufacturing.

S. Yadav is a graduate student in the Department of Mechanical and Manufacturing Engineering, University of Calgary pursuing MSc degree. He received his BSc degree in Mechanical Engineering from Allahabad University in 1996. His research interests include knowledge base and database representation for intelligent CAD/CAM systems, 3D geometric modeling, and product life-cycle modeling.